Map (dictionary)

- Store a key-value pair (K, V)

find(K)
put(K, V)
erase(K)
at(K)

insert(5, C)
insert(10, D)
insert(12, B)
insert(2, A)

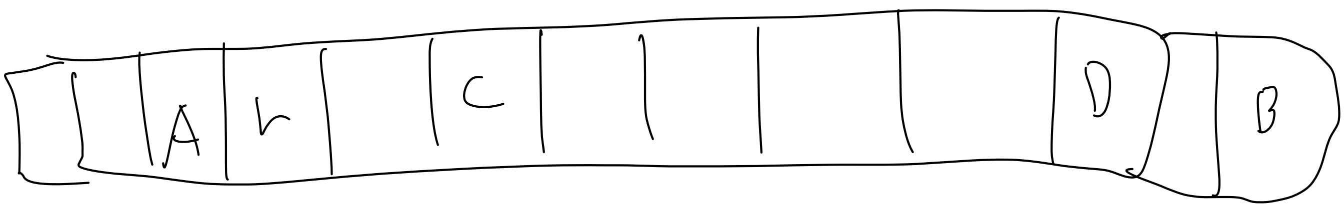
at(5) = C

find(2) = end

linked list

(5, C) → (10, D) → (12, B) → (2, A) → (10000, h)

insert : O(1)
find : O(n)



insert(10000, h) =

Hash Table

is Prime

$h(x) = x \bmod \boxed{N}$

at(10) => 10 % 5 = 0 => H

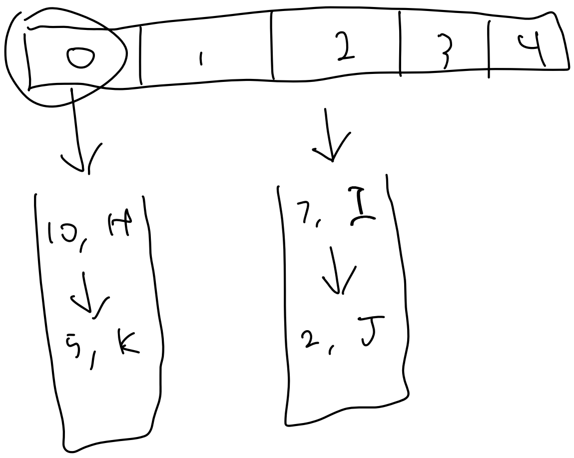
(10, H) => 10 % 5   0

(7, I) => 7 % 5   2

(2, J) => 2 % 5   2

(5, K) => 5 % 5   0

Chaining



find: O(n)

O(1) *

$i = K \bmod N$

linear probing

$A[i + 1 \bmod N]$

$(10, H) \Rightarrow 10 \% 5 = 0$

$(7, I) \Rightarrow 7 \% 5 = 2$

$(2, J) \Rightarrow 2 \% 5 = 2$

$(4, K) \Rightarrow 4 \% 5 = 4$

$(9, Q) \Rightarrow 9 \% 5 = 4$

0	1	2	3	4
H	Q	I	J	K

J   Q →

quadratic Probing

$i = K \bmod N$

$A[i + j^2 \bmod N]$ $\quad j = 0, 1, 2, 3 \ldots$

$j = 0, 1, 4, 9 \ldots$

$(10, H) \Rightarrow 10 \% 5 = 0$

$(7, I) \Rightarrow 7 \% 5 = 2$

$(2, J) \Rightarrow 2 \% 5 = 2$

$(4, K) \Rightarrow 4 \% 5 = 4$

$(9, Q) \Rightarrow 9 \% 5 = 4$

0	1	2	3	4
H	Q	I	J	K
		J		Q

1. What is the worst-case running time for inserting n key-value entries into initially empty map M that is implemented with a list? $O(n)$

Draw the 11-entry hash table that results from using the hash function, h(k) = (3k+3) mod 11, to hash keys: 12, 44, 13, 88, 23, 94, 11, 39, 5, 20, and 16

a. Collisions are handled by chaining
b. Collisions are handled by linear probing
c. Collisions are handled by quadratic probing, up to the point where the method fails

12 → 39 % 11 = 6
44 → 3
13 → 42 % 11 = 9
88 → 3
23 → 6
94 → 10
11 → 36 % 11 = 3
39 → 10
5 → 18 % 11 = 7
20 → 8
16 → 7

a) chaining

0	1	2	3	4	5	6	7	8	9	10
			↓			↓	↓	↓	↓	↓
			44			12	5	20	13	94
			↓			↓	↓			↓
			88			23	16			39
			↓							
			11							

b) linear Probing

0	1	2	3	4	5	6	7	8	9	10
39	20	16	44	88	11	12	23	5	13	94

88 23 39
11 5 20
16

c) quadratic probing

0	1	2	3	4	5	6	7	8	9	10
39	20		44	88	11	12	23	5	13	94

88 23 39
11 5 20
16

3K+3 % 11

12 → 39 % 11 = 6

44 → 3

13 → 9

88 → 3

23 → 6

94 → 10

11 → 3

39 → 10

5 → 7

20 → 8

16 → 7

0	1	2	3	4	5	6	7	8	9	10
			44			12	5	20	13	94
			88			23	16			39
			11							

0	1	2	3	4	5	6	7	8	9	10
39	20	16	44	88	11	12	23	5	13	94

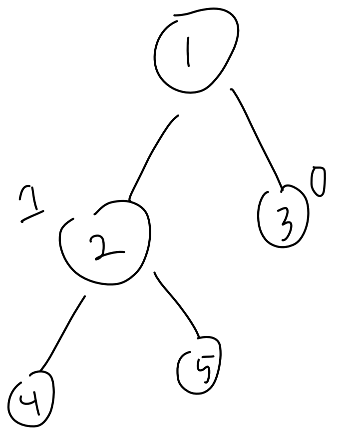
88
11
23
5
39
20
16

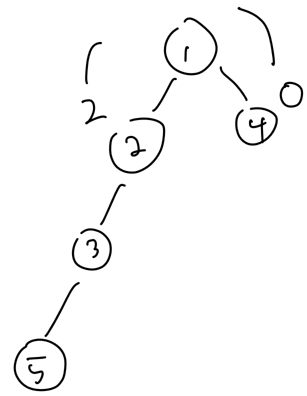
0	1	2	3	4	5	6	7	8	9	10
39	20		44	88	11	12	23	5	13	94

88
23
39
5
11
20
16

Binary Trees

- balanced: left and right sustree's height difer by at most 1



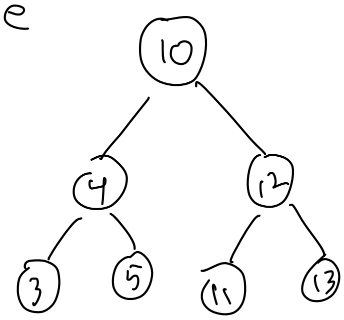


Binary Search Tree

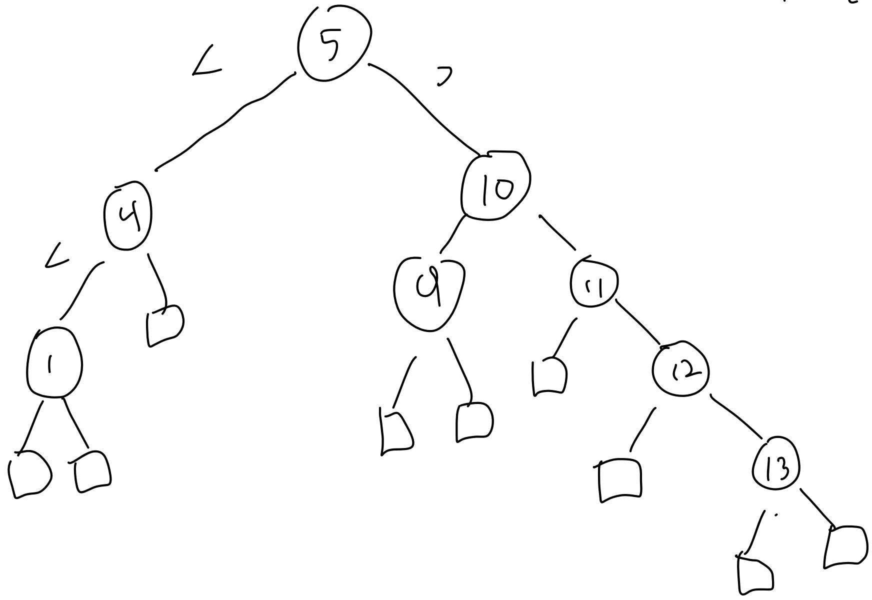
put(k, v)

put(1, v)

erase(3)



inorder: 3, 4, 5, 10, 11, 12, 13



put() : O(n)

erase() : O(n)

find() : O(n)

AVL Tree

self-balancing

insert()	$O(\log(n))$
delete()	$O(\log(n))$
find()	$O(\log(n))$